Assignment 9: Neural Network with TensorFlow for MNIST Classification

Shiva Agarwal

May 16, 2024

Assignment Questions

Question 1: Importing Libraries

1. Import the required machine learning libraries for the analysis.

Question 2: Reading the MNIST Dataset

1. Read the MNIST dataset stored in ubyte format and store it as a variable using the TensorFlow MNIST library.
2. Mention the location where the datasets would be stored. Use `MNIST/` for this assignment.
3. Download the data and ensure the datasets are unpacked correctly.

Question 3: Defining Parameters

1. Define the various deep learning parameters: image size, number of labels, learning rate, number of steps, and batch size.

Question 4: Defining Placeholders

1. Define a TensorFlow placeholder for the training dataset (`x_train`) and the labels (`y_train`).
2. Print the training set placeholder variable to ensure it has been assigned correctly.

Question 5: Visualizing the Dataset

1. Print the first few digits present in the dataset.
2. Reshape the images into 28x28 pixels and plot them using the matplotlib library.

Question 6: Creating Placeholders and Variables for the Neural Network

1. Create placeholders and variables for the input layer, weights, bias, and output layer.

Question 7: Defining the Model

1. Apply the softmax activation function to create the output layer.
2. Create a placeholder to input the correct answers.
3. Use cross-entropy as a measure to understand the precision of the model.

Question 8: Training the Model

1. Minimize the cross-entropy using the Gradient Descent Optimizer.
2. Execute the commands to train the dataset for 1000 iterations.

Question 9: Evaluating the Model

1. Print the accuracy of the model based on the predictions made by the neural network.

Solution

Question 1: Importing Libraries

```
1 import tensorflow as tf
2 from tensorflow.examples.tutorials.mnist import
      input_data
3 import numpy as np
4 import matplotlib.pyplot as plt
```

Question 2: Reading the MNIST Dataset

```
1 # Read the data from the TensorFlow in-built mnist
      dataset
2 mnist = input_data.read_data_sets("MNIST/", one_hot=True)
```

Question 3: Defining Parameters

```
1 # Define the various Deep Learning parameters as well as
      the image and label size
2 image_size = 28
3 num_labels = 10
4 learning_rate = 0.05
5 number_of_steps = 1000
6 batch_size = 100
```

Question 4: Defining Placeholders

```
1 # Define placeholders
2 x_train = tf.placeholder(tf.float32, [None, image_size *
      image_size])
3 y_train = tf.placeholder(tf.float32, [None, num_labels])
4
5 # Print the training set placeholder variable
6 print(x_train)
```

Question 5: Visualizing the Dataset

```
1 left = 2.5
2 top = 2.5
3 fig = plt.figure(figsize=(10,10))
```

```
for i in range(6):
    ax = fig.add_subplot(3,2,i+1)
    im = np.reshape(mnist.train.images[i,:], [28,28])
    label = np.argmax(mnist.train.labels[i,:])
    ax.imshow(im, cmap='Greys')
    ax.text(left, top, str(label))
plt.show()
```

Question 6: Creating Placeholders and Variables for the Neural Network

```
# A placeholder for the data (inputs and outputs)
inputs = tf.placeholder(tf.float32, [None, 784])
# Weights: the weights for each pixel for each class
# bias: bias of each class
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))
```

Question 7: Defining the Model

```
# The model
outputs = tf.nn.softmax(tf.matmul(inputs, Weights) + bias
    )

# Create a placeholder to input the right answers
y_ = tf.placeholder(tf.float32, [None, 10])
# A measure of model precision using cross-entropy
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log
    (outputs), reduction_indices=[1]))
```

Question 8: Training the Model

```
# Minimize the resulting cross_entropy using gradient
    descent
train_step = tf.train.GradientDescentOptimizer(
    learning_rate).minimize(cross_entropy)

# Initialize the variables
init = tf.global_variables_initializer()

# Create a session
sess = tf.Session()
```

```
sess.run(init)

# Run training step 1000 times
for i in range(number_of_steps):
    # Get random 100 data samples from the training set
    batch_xs, batch_ys = mnist.train.next_batch(
        batch_size)
    # Feed them to the model in place of the placeholders
        defined above
    sess.run(train_step, feed_dict={inputs: batch_xs, y_:
        batch_ys})
```

Question 9: Evaluating the Model

```
# Evaluate the model
correct_prediction = tf.equal(tf.argmax(outputs, 1), tf.
    argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.
    float32))

# Print the accuracy
print("Accuracy:", sess.run(accuracy, feed_dict={inputs:
    mnist.test.images, y_: mnist.test.labels}))
```